Design and Implementation of the Autonomous Underwater Vehicle: *Zoidberg* San Diego City Robotics

Anna Kelley, Arnold Suarez, Benny De La Rosa, Bob Pruitt, Charles A., Christian Pratt, DJ Brown, Gina Bochicchio, Jackie Alcazar, Jasmeet R., JoAna Smith, Leila Firestone, Matt Dick, Melissa Bodenmiller, Ned Richards, Nikita Bobrov, Oskar Garcia, Seeraj S., Thomas Lucio, Tomas Mendoza

Abstract - Zoidberg is an Autonomous Underwater Vehicle (AUV) built by San Diego City Robotics (SDCR) for the 2019 intercollegiate RoboSub competition. Utilizing the previous competition's experiences, successes and failures, Zoidberg is an improved, enhanced and more refined version of itself, built by approximately 30 community college students, with the support of advisors and faculty. These advancements in Zoidberg's design include decisive implementation of new mission control, acoustic and vision software, as well as a new killswitch, constructed for improved reliability at the conclusion of a run. SDCR looks to vastly improve their place in the RoboSub competition as Zoidberg is exceedingly capable of a top finish.

I. COMPETITIVE STRATEGY

SDCR's competitive strategy for Robosub 2019 has been to improve the robustness and reliability of the Zoidberg robot, with the aim of accomplishing simple tasks. This has consisted of developing the control system, including hardware, and basic software framework to be straightforward; the goal of this development is to increase the usability of the systems by team members, and ensure its dependability among users, and from year-to-year. The major changes consist of improving the kill switch and replacing Robotic Operating System (ROS) with Python. With these improvements, SDCR's goal is to reliably participate in all trial runs at the competition, and maximize opportunities for competition scoring, as well as any necessary troubleshooting of other robot components.

The most significant hardware change this year is an improved kill switch to replace a faulty design that resulted in the team having to forego multiple competition runs during last year's competition. Mechanically rotating the switch to turn Zoidberg on or off, the switch has an operating voltage and operating current of 120 Volts and 5 Amps, respectively. The new kill switch design will maximize pool time, both for trials and scoring, and therefore improve Zoidberg's chances of placing into the finals.

The largest software development was to remove the Robotic operating system

(ROS) completely. Although ROS provides many powerful capabilities, it has required significant effort to install and learn; this is difficult to accomplish for instructors and users while there are other concurrent robot development and testing tasks. In the past, this has limited the software users and developers to a small core group, and made the software difficult for others.

This year's new software framework is written entirely in the open-source programming language Python. By focusing on a small subset of behaviors required for the competition, this development has focused on reliability. Since there are new team members every year, increasing the software framework's ease-of-use is important for ensuring its reliability, both among users and from year-to-year.

The development of the Python-based control system required a significant investment of team member time, but was made accessible to multiple users (including those specializing in other robot components). This development resulted in a software framework that is modular, so that the integrated robotic system can be evaluated based on its individual parts. This modular design ensures that robot testing can be conducted in a component-wise manner; this robust capability has allowed the team to better test the Zoidberg robot. An example of this is the ability of the vision team to bench-test ZED camera's depth sensing capability, which will be used to identify and touch the buoys in the competition. The modular software therefore also allows for parallel testing of robot components, prior to full-integration during in-water testing.

In both improving the kill-switch and migrating the software framework from ROS to Python, we have prioritized reliability over complexity. This has allowed for significant improvements in our testing-capabilities, as well as increasing the participation of team members in control system and troubleshooting tasks. These changes have also led to greater in-water testing time during the year, and promise more reliable participation of the Zoidberg robot at all competition trials, and therefore greater opportunities for scoring.

II. VEHICLE DESIGN

A. Mechanical

A newly constructed killswitch was implemented for Zoidberg, relying on mechanical movement as opposed to exploiting magnetism for the previous killswitch. Resulting in further opportunities of increasing the AUV's reliability, Zoidberg will be capable of accessing and completing a broader range of tasks in this year's competition.

Newly designed torpedos were designed for Zoidberg specifically for the "Stake in the Heart" task, where the torpedos themselves are 3D printed, while 1.5 inch long PVC pipe serve as individual air tanks.

B. Electrical

The majority of Zoidberg's electrical hardware was left the same for this year's competition due to proven success of the system last year. With the elimination of the hall effect kill switch, the electrical hardware was simplified, while the rest of its components had minimal modifications.

C. Software

All of Zoidberg's software is written in Python due to its efficiency, readability and effectiveness. For navigation and mission control, self sufficient code handles every process previously controlled by ROS; a huge step for Zoidberg, as independency and workflow was limited when working with ROS.

Zoidberg is stabilized underwater by the Pixhawk motor controller, which communicates with the onboard TX1 to fire the AUV's motors for particular movements. Pixhawk's main functionality resides in conjunction with drones for ArduSub [1].

Vision software utilizes the OpenCV computer vision library for Zoidberg's object detection algorithms. Written in Python, Zoidberg's vision proficiency provides a significant advantage during the competition, as the detection algorithms utilizing OpenCV's image analyzation functions supplies Zoidberg's mission control software with valuable information to make decisions. For a given task, the algorithms will examine every frame the ZED Camera captures and produce necessary coordinates for Zoidberg to make informed and conclusive decisions whether to navigate through a gate *(fig 1)*, drive up to a specific monster and bump the buoy or where to fire a torpedo.

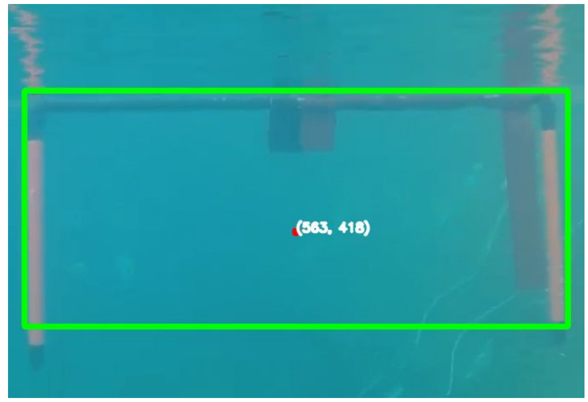

Fig 1. Navigating through a gate.

III. EXPERIMENTAL RESULTS

For the 2019 RoboSub competition, an opportunity arose for the team to consistently test Zoidberg in a water tank on San Diego City College's campus, approximately 4 feet long, 8 feet wide and 8 feet deep, a first in SDCR's time competing, as local swimming pools were the default medium of all testing. Zoidberg's software was tested extensively on the bench-top utilizing the ZED camera for capturing footage to further optimize vision's object detection algorithms.

IV. ACKNOWLEDGEMENTS

SDCR thanks all individual team members for consistently contributing your time, efforts and enthusiasm for this year's competition. We thank Gina Bochicchio, our faculty advisor for this semester, for her contributions. We also like to thank all

previous SDCR club members for their continued and ongoing support. SDCR acknowledges all new-comers as they will receive the torch for next season to continue on with Zoidberg's quest to place first in the RoboSub competition.

V. REFERENCES

[1] ArduPilot, "Pixhawk Overview", ArduPilot, http://ardupilot.org/copter/docs/common-pixhawk-overview.html

Appendix A: Expectations

Subjective Measures

	Maximum Points	Expected Points	Points Scored
Utility of Team Website	**50**	50	
Technical Merit	**150**	150	
Written Style	**50**	50	
Capability for Autonomous Behavior	**100**	100	
Creativity in System Design	**100**	80	
Team Uniform	**10**	5	
Team Video	**50**	50	
Pre-Qualifying Video	**100**	100	
Discretionary Points	**40**	40	
Total	**650**	630	

Performance Measures

	Maximum Points	Expected Points	Points Scored
Weight	**See Table 1 / Vehicle**		
Marker/Torpedo overweight or size <10%	**minus 500 / marker**		
Gate: Pass Through	**100**	100	
Gate: Maintain Fixed Heading	**150**	150	
Gate: Coin Flip	**300**	150	
Gate: Pass through 60% section	**200**	150	
Gate: Pass through 40% section	**400**	200	
Gate: Style	**+100 (8x max)**	100	

Collect Pickup: Crucifix, Garlie	**400 / object**	0	
Follow the "path" (2 total)	**100 / segment**	0	
Slay Vampire: Any, Called	**300, 600**	600	
Drop Garlic: Open, Closed	**700, 1000 / marker (2+ pickup)**	0	
Drop Garlic: Move Arm	**400**	0	
Stake through Heart: Open Oval, Cover Oval, Sm Heart	**800, 1000, 1200 / torpedo (max 2)**	800	
Stake through Heart: Move lever	**400**	0	
Stake through Heart: Bonus - Cover Oval, Sm Heart	**500**	0	
Expose to Sunlight: Surface in Area	**1000**	500	
Expose to Sunlight: Surface in Object	**400 / object**	0	
Expose to Sunlight: Open Coffin	**400**	0	
Expose to Sunlight: Drop Coffin	**200 / object (Crucifix only)**	0	
Random Pinger first task	**500**	500	
Random Pinger second task	**1500**	1500	
Inter-vehicle Communication	**1000**	0	
Finish the mission with T minutes (whole + fractional)	**Tx100**	0	

Appendix B: Component Specifications

Component	Vendor	Model/Type	Specs	Cost (if new)
Buoyancy Control				
Frame	Used	Custom Fabrication	HDPE	
Waterproof Housing	BlueRobotics	6” Water Tight	Additional End Caps	
Waterproof Connectors	TE Connectivity	Wet-Conn	Mini	
Thrusters	BlueRobotics	T200		
Motor Control	MRobotics	Rev2		
High Level Control				
Actuators				
Propellers				
Battery	Turnenergy	LiPo	11.1 Vo, 5.5 A Charging Current	
Converter				
Regulator				
CPU	NVidia	Jetson TX1		
Internal Comm Network				
External Comm Interface	Serial			
Programming Language	Python			

Compass	Pixhawk			
Inertial Measurement Unit (IMU)				
Doppler Velocity Log (DVL)				
Camera(s)	StereoLabs	ZED	Stereo Camera	
Hydrophones	Used			
Manipulator	N/A			
Algorithms: Vision	OpenCV			
Algorithms: Acoustics	Python			
Algorithms: Localization and Mapping	Python			
Algorithms: Autonomy	Python and OpenCV			
Open Source Software				
Team Size	30			
HW/SW Expertise Ratio	10:1			
Testing Time: Simulation	~40 hrs			
Testing Time: In-Water	~5 hrs			